



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/073,851	02/11/2002	Grzegorz J. Czajkowski	004-7056	8021

42714 7590 06/26/2006

ZAGORIN O'BRIEN GRAHAM LLP (004)
7600B NORTH CAPITAL OF TEXAS HIGHWAY
SUITE 350
AUSTIN, TX 78731-1191

EXAMINER

BULLOCK JR, LEWIS ALEXANDER

ART UNIT	PAPER NUMBER
2195	

DATE MAILED: 06/26/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary

Application No.

10/073,851

Applicant(s)

CZAJKOWSKI ET AL.

Examiner

Lewis A. Bullock, Jr.

Art Unit

2195

– The MAILING DATE of this communication appears on the cover sheet with the correspondence address –
Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 10 April 2006.
- 2a) ☐ This action is **FINAL**. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 26-32, 36-45 and 47-53 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) _____ is/are allowed.
- 6) ☒ Claim(s) 26-32, 36-38, 40-45 and 47-53 is/are rejected.
- 7) ☒ Claim(s) 39 is/are objected to.
- 8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on _____ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some * c) ☐ None of:
- ☐ Certified copies of the priority documents have been received.
 - ☐ Certified copies of the priority documents have been received in Application No. _____.
 - ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- ☒ Notice of References Cited (PTO-892)
- ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
- ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)
Paper No(s)/Mail Date _____.
- ☐ Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____.
- ☐ Notice of Informal Patent Application (PTO-152)
- ☐ Other: _____.

DETAILED ACTION

Claim Rejections - 35 USC § 103

1. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

2. Claims 26-32, 36-38, 40-45, and 47-53 are rejected under 35 U.S.C. 103(a) as being unpatentable over DRAVES (U.S. Patent 6,349,355).

As to claim 26, DRAVES teaches a method for handling sharing of a physical memory space between a first process and a second process, the method comprising: allocating a portion of the physical memory space (col. 12, lines 41-55) and creating a buffer object (memory reference / address mapping) responsive to a buffer object request (request for access to a page or module) (col. 11, line 61 – col. 12, line 35; col. 12, lines 56 – col. 13, line 7; col. 13, lines 20-33) wherein one of the processes executes native code (kernel function) of a program and the other process executes user language code (user program) of the program; and mapping a first address range of the first process and a second address range of the second process to the allocated portion of the shared physical memory space, wherein the buffer object represents at least one of the address ranges (virtual address ranges / address mapping) (col. 11, line 61 – col. 12, line 35; col. 12, lines 56 – col. 13, line 7; col. 13, lines 20-33). However, DRAVES does not teach that the user program is a safe language code. Official Notice is taken in that Java is a well known safe language user code program / thread and

therefore it would be obvious to one of ordinary skill in the art that a Java user program / thread invokes a kernel function as defined by DRAVES to shared executable modules between kernel and user threads (col. 4, lines 52-57).

As to claim 42, DRAVES teaches a computer program encoded on one or more computer readable media (col. 6, lines 7-26), the computer program comprising: a first language code executable to allocate an address range in a first environment, which executes the first language code, responsive to a request for a buffer object, to map the first environment address range to a portion of a memory space, to create the buffer object as representative of the allocated address range, and to communicate the buffer object and the portion of memory space to another environment that executes a different language code (via the user program creating an address range / mapping for the user program and sharing it with the kernel function to thereby share an address range via passing a memory reference to the kernel function) (col. 11, line 61 – col. 12, line 35; col. 12, lines 56 – col. 13, line 7; col. 13, lines 20-33); and a second language code executable to request the buffer object and to allocate an address range in a second environment, which executes the second language code, and executable to map the second environment address range to the portion of the memory space (via the kernel function mapping the address range with the user range and calling the program module or dereferencing the memory) (col. 11, line 61 – col. 12, line 35; col. 12, lines 56 – col. 13, line 7; col. 13, lines 20-33), wherein one of the first and second language codes comprises a user language code (user process) and the other of the first second

Art Unit: 2195

language codes comprises a native code (kernel function). However, DRAVES does not teach that the user program is a safe language code. Official Notice is taken in that Java is a well known safe language user code program / thread and therefore it would be obvious to one of ordinary skill in the art that a Java user program / thread invokes a kernel function as defined by DRAVES to shared executable modules between kernel and user threads (col. 4, lines 52-57).

As to claim 47, DRAVES teaches an apparatus (computer system) comprising: a memory (col. 5, line 66 – col. 6, line 34); and means for mapping an address range in a first process for a first code (user address range) and an address range in a second process for a second code (kernel address range) to a same region of the memory to facilitate transparent code isolation (via the kernel function mapping the address range with the user range and calling the program module or dereferencing the memory) (col. 11, line 61 – col. 12, line 35; col. 12, lines 56 – col. 13, line 7; col. 13, lines 20-33), and for representing at least the memory to facilitate transparent code isolation, and for representing at least one of the address ranges with a buffer object (address mapping / reference to program module) (col. 11, line 61 – col. 12, line 35; col. 12, lines 56 – col. 13, line 7; col. 13, lines 20-33), wherein one of the first and second processes executes native code (kernel function) and the other of the first and second processes executes user language code (user process). However, DRAVES does not teach that the user program is a safe language code. Official Notice is taken in that Java is a well known safe language user code program / thread and therefore it would be obvious to one of

Art Unit: 2195

ordinary skill in the art that a Java user program / thread invokes a kernel function as defined by DRAVES to shared executable modules between kernel and user threads (col. 4, lines 52-57).

As to claim 50, DRAVES teaches a method of performing native code isolation in the presence of direct buffers without losing transparency, the method comprising: mapping a first address range of a first process (user process virtual address range) and a second address range of a second process (kernel function virtual address range) to a same portion of memory space (via the kernel function mapping the address range with the user range and calling the program module or dereferencing the memory) (col. 11, line 61 – col. 12, line 35; col. 12, lines 56 – col. 13, line 7; col. 13, lines 20-33), wherein one of the first and second processes executes native code (native function) and the other of the first and second processes executes a user language code (user process); and creating a direct buffer (address mapping / reference to program module) (col. 11, line 61 – col. 12, line 35; col. 12, lines 56 – col. 13, line 7; col. 13, lines 20-33) to represent at least one of the first and the second address ranges (via using the mapping / reference for either the kernel function / user process to access the address range of the other component or programs of the other component). However, DRAVES does not teach that the user program is a safe language code. Official Notice is taken in that Java is a well known safe language user code program / thread and therefore it would be obvious to one of ordinary skill in the art that a Java user program /

Art Unit: 2195

thread invokes a kernel function as defined by DRAVES to shared executable modules between kernel and user threads (col. 4, lines 52-57).

As to claim 27, DRAVES teaches at least one of the address ranges comprises virtual addresses (col. 7, lines 3-6).

As to claim 28, DRAVES teaches the physical memory space comprises a set of one or more physical pages (col. 11, line 61 – col. 12, line 35; col. 12, lines 56 – col. 13, line 7; col. 13, lines 20-33).

As to claim 29, DRAVES teaches the second process creating the buffer object and indicating the allocated physical memory space portion and the buffer object to the first process (via passing a memory reference to the other process) (col. 11, line 61 – col. 12, line 35; col. 12, lines 56 – col. 13, line 7; col. 13, lines 20-33).

As to claim 30, Official Notice is taken in that Java is a well known safe language user code program / thread and therefore it would be obvious to one of ordinary skill in the art that a Java user program / thread invokes a kernel function as defined by DRAVES to shared executable modules between kernel and user threads (col. 4, lines 52-57).

As to claim 31, DRAVES teaches the first process indicating the buffer object to one or more callers (via sending the reference/mapping to the kernel function) (col. 11, line 61 – col. 12, line 35; col. 12, lines 56 – col. 13, line 7; col. 13, lines 20-33).

As to claim 32, DRAVES teaches maintaining an encoding that indicates allocated portions of the physical memory space to allow detection of at least one of overlapping address ranges and nested address ranges (mappings stored in a conventional translation lookaside buffer wherein the virtual addresses overlap) (col. 7, lines 35-37).

As to claims 43 and 44, DRAVES teaches communication between the user level and the kernel level (col. 11, line 61 – col. 12, line 35; col. 12, lines 56 – col. 13, line 7; col. 13, lines 20-33). However, DRAVES does not teach an interface between the two levels. Official Notice is taken in that it is well known in the art that an interface must be used when communicated from the user level to the kernel level, i.e. a gate function or JNI and therefore obvious to one of ordinary skill in the art that when a user process calls or sends a reference to a kernel function as detailed in DRAVES, it invokes the interface. It would also be obvious that the user/kernel environment is on a virtual machine since DRAVES allows for the computer system to be any one of a variety of different types of devices (col. 6, lines 1-34) and virtual machines are well known in the art.

As to claim 45, DRAVES teaches the other of the first and second environments that executes native code comprises an operating system (col. 6, lines 35-45).

As to claim 53, DRAVES teaches the second language code is further executable to indicate the buffer object to a caller (via passing a memory reference/mapping to the other process/function) (col. 11, line 61 – col. 12, line 35; col. 12, lines 56 – col. 13, line 7; col. 13, lines 20-33).

As to claim 48, DRAVES teaches means for communicating an identifier of the buffer object between the first and second processes (via passing a memory reference/mapping to the other process/function) (col. 11, line 61 – col. 12, line 35; col. 12, lines 56 – col. 13, line 7; col. 13, lines 20-33).

As to claim 49, DRAVES teaches means for detecting at least one of nested address ranges and overlapping address ranges (mappings stored in a conventional translation lookaside buffer wherein the virtual addresses overlap) (col. 7, lines 35-37).

As to claim 36, DRAVES teaches communicating an identifier of the created direct buffer to one or more callers (via passing a memory reference/mapping to the other process/function) (col. 11, line 61 – col. 12, line 35; col. 12, lines 56 – col. 13, line 7; col. 13, lines 20-33).

As to claim 37, DRAVES teaches the memory space comprises a set of one or more physical memory pages (col. 11, line 61 – col. 12, line 35; col. 12, lines 56 – col. 13, line 7; col. 13, lines 20-33).

As to claim 38, DRAVES teaches allowing address ranges in the first process to overlap and/or nest (via the virtual address ranges of the user process and kernel function overlap) (col. 11, line 61 – col. 12, line 35; col. 12, lines 56 – col. 13, line 7; col. 13, lines 20-33).

As to claim 40, DRAVES teaches maintaining a list that indicates mapped address ranges to allow detection of at least one of overlapping address ranges and nested address ranges (mappings stored in a conventional translation lookaside buffer wherein the virtual addresses overlap) (col. 7, lines 35-37).

As to claim 41, DRAVES teaches at least one of the first process address range and the second address range comprises virtual addresses (col. 7, lines 3-6).

As to claim 51, DRAVES teaches the direct buffer is created by the second process responsive to receiving a direct buffer request from the first process (via passing a memory reference/mapping to the other process/function) (col. 11, line 61 – col. 12, line 35; col. 12, lines 56 – col. 13, line 7; col. 13, lines 20-33).

As to claim 52, DRAVES teaches the same portion of memory space is allocated (wherein the virtual address ranges of the kernel function / user process are the same) (col. 11, line 61 – col. 12, line 35; col. 12, lines 56 – col. 13, line 7; col. 13, lines 20-33) from one of a memory mapped file and a frame buffer (via the memory includes floppy disks, and other types of computer-readable storage) (col. 6, lines 18-26).

Allowable Subject Matter

3. Claim 39 is objected to as being dependent upon a rejected base claim, but would be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims.

Response to Arguments

4. Applicant's arguments with respect to claims 26-32, 36-38, 40-45, and 47-53 have been considered but are moot in view of the new ground(s) of rejection.


Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Lewis A. Bullock, Jr. whose telephone number is (571) 272-3759. The examiner can normally be reached on Monday-Friday, 8:30 a.m. - 5:00 p.m..

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng An can be reached on (571) 272-3756. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

June 21, 2006


LEWIS A. BULLOCK, JR.
PRIMARY EXAMINER